NETWORK HEADER COMPRESSION ARRANGEMENT

BACKGROUND OF THE INVENTION

The present invention pertains to network internet protocol and more particularly to a header compression arrangement for wireless network applications.

A prior art method of sending sequencing and timing information to support real time services in internet protocol (IP) networks is by use of the real time protocol (RTP). Network transport for voice is provided by UDP (User Data Protocol) protocol. Wireless IP network applications use vocoded voice. Each vocoded frame is typically 10 to 20 bytes in length of information. The combined RTP/UDP/IP header that is therefore required to be attached to the packet so that the voice frame can reach its destination has a length of about 40 bytes.

The 40 byte combined header is considerable overhead with respect to about 20 bytes of actual voice information for typical wireless vocoder. Sending the combination of 40 bytes of header information with 20 bytes of voice information constitutes an inefficient use of a radio frequency (RF) link.

One approach to the overhead problem is to pack several voice frames together with a single header to improve RF transmission efficiency. However, this approach increases the voice delay significantly. For example, in a CDMA 2000 network, each additional voice frame increases the end to end delay by approximately 60 milliseconds. This produces an unacceptable voice quality.

Voice quality may be further sacrificed to gain bandwidth by using lower bit-rate modes of the vocoder. Again, this is an undesirable situation.

Another approach to the overhead problem is header compression. Available header compression mechanisms are typically designed for generic data compression. Such generic data compression does not take advantage of the peculiar characteristics of voice over packet data. As a result, such header compression techniques perform sub optimally.

Accordingly, it would be highly desirable to have a header compression arrangement for a wireless IP network which makes efficient use of the RF link while providing high quality voice data throughout the network.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 is a block diagram of a header compression arrangement in accordance with the present invention.

FIG. 2 is a layout of a UDP header from the prior art.

FIG. 3 is a layout of an RTP header from the prior art.

FIG. 4 is a layout of a compressed UDP/RTP header from the prior art.

FIG. 5 is a layout of a TCP/IP header from the prior art.

FIG. 6 is a layout of a Van Jacobson compressed TCP/IP header from the prior art.

FIG. 7 is a layout of a compressed header in accordance with the present invention.

FIG. 8 is a flowchart of mobile station header compression in accordance with the present invention.

FIG. 9 is a flowchart of mobile station header decompression in accordance with the present invention.

FIG. 10 is a flowchart of packet data service node header compression in accordance with the present invention.

FIG. 11 is a flowchart of packet data service node header decompression in accordance with the present invention.

DESCRIPTION OF THE PREFERRED EMBODIMENT

Referring to FIG. 1, a block diagram of a header compression/decompression arrangement in accordance with the present invention is shown. The present invention pertains to voice over internet protocol (IP) for wireless networks, such as 3G and more advanced technology networks which use vocoded voice. Header compression arrangement 100 includes a mobile station 15 and a Packet Data Service Node 55. The mobile station 15 includes a robust header compression (ROHC) based RTP/UDP header compression/decompression unit 10. Header compression/decompression is well known in the art and may be found at IETF RFC 3095 published July 2001.

Referring to FIG. 2, A prior art UDP header is shown. The cross-hatch fields of this header, namely source, destination port and length stay constant and are not required to be sent often. A checksum for the header is different for each data set.

Referring to FIG. 3, a prior art RTP header 120 is shown. As with the UDP header 110, the RTP header 120 has crossed hatch fields which need not be sent every time. Block 10 of FIG. 1 compresses the RTP header to the form shown in FIG. 4.

FIG. 4 depicts the results of the robust header compression of block 10 of FIG. 1. The result of the robust header compression 10 is the three byte data structure 130 shown in FIG. 4. This result includes the UDP checksum (2 bytes) and a summary byte indicating field changes for the RTP portion of the header. A bit of the byte pertains to each of the fields for the RTP header 120.

In order to transmit the data packet through the internet protocol portion of the network, the Van Jacobson TCP/IP header compression/decompression 20 is performed. A prior art TCP/IP header 140 is shown in FIG. 5. The Van Jacobson TCP/IP header compression/decompression 20 is well known in the art and was published in the IETF RFC 1144 on February, 1990. Again, the cross-hatched information in the typical TCP/IP header 140 does not change and TCP/IP header compression/decompression 20 does not transmit these after the first packet of data is sent. The TCP/IP header 140 is typically 40 bytes in length. After the compression/decompression 20 is performed, the result is shown in data structure 150 of FIG. 6.

The Van Jacobson TCP/IP header compression further noted that the two byte total length field of data structure 140 and the two byte IP header checksum field were not necessary. Therefore the Van Jacobson method 20 deleted these.

The Van Jacobson compressor 20 sends uncompressed headers at the beginning of a transmission and in order to resynchronize with decompresses upon detection of corrupted header packets. Otherwise, data structure 150 of FIG. 6 is sent. This data structure includes a mask byte 151. The mask byte 151 indicates which of the data fields in the remainder of the data structure has changed. A "one" may be used to indicate the change. Further, a TCP checksum 152 is included in the data structure 150. Also included are six other parameters from the typical TCP/IP header of 140 of FIG. 5.

Since the receiving end and the transmitting end of the header compression arrangement 100 may be either a mobile station or a Packet Data Service Node (PDSN) both the PDSN 55 and the mobile station 15 must include the compression and decompression methodology. That is both the network and the mobile station must be able to transmit and receive voice data and therefore are required to have the compressor and decompressor. In the PDSN 55, a compressor/decompressor 50 performs a decompression function decompresses an IP (internet protocol) portion of the header for processing throughout the wireline portion of the wireless internet.

The receiving end of the voice data packet, whether it is the PDSN or mobile station, saves a copy of the last non-compressed header for each flow along with a flow identifier. Further, for each of the compressed headers that are sent from the transmitting end to the receiving end, the decompression methodology regenerates the IP header checksum. Further, the decompressor fills in the remaining bytes saved from the last non-compressed TCP/IP header.

The robust header compressor 10 transmits three bytes to header assembly 40. Typically Van Jacobson compressor 20 would transmit the nine byte data structure 150 to header assembly 40.

The present invention includes new or ancillary TCP header 30. New TCP header 30 uses the standard Van Jacobson TCP/IP header compression method to compress IP headers through a radio access network. Ancillary TCP header 30 adds a "new" TCP header to each internet protocol header.

The new or ancillary TCP/IP header 160 is shown in FIG. 7. It includes a first byte 161, a connection identification 163, and a TCP checksum of two bytes 162. The specific bit values in the first byte 161 are shown. The leftmost bit is always fixed and therefore is not of concern to this methodology. The other seven bits have a specific value of zero or one. The rightmost four bits of byte 161 being set equal to one, as shown in FIG. 7, indicate to the Van Jacobson method 20 that the encoding is to be performed for a special case, that being a unidirectional data transfer.

The connection identification (id) 163 is optional to the extent that if the physical connection supports a single packet flow, the connection id does not have to be resent.

Standard gateway equipment (not shown) which supports the Van Jacobson methodology believes that it is dealing with a normal TCP/IP type connection. Only the end points such as the mobile station and PDSN are aware that this is not a typical TCP/IP connection and involves a new or ancillary TCP header.

In the data structure 160 of FIG. 7 the bits indicate, for a zero no change in the particular field and for one a change in the particular field. The four rightmost bits of byte 161 all being set to one indicate the unidirectional data transfer special case. Since the new TCP header indicates a constant sequence number field change, the four bits from the right being always set, the change is always the same and therefore the sequence field is not transmitted as part of the header.

For the Van Jacobson methodology the specific bit values in the first byte 161 disables the error recovery procedure, also known as "resynchronization," of the Van Jacobson method 20. That is, the error recovery is not needed for voice packets since voice processing applications at the network ends simply discard any corrupted voice packets. Further, when the new TCP header is being decompressed, it is not discarded until the TCP protocol header is reconstructed, that is added back in at the receiving end to form the uncompressed (or original) TCP header.

As a result of using the new TCP header arrangement, further IP header compression is obtained over the Van Jacobson methodology by using the Van Jacobson method in a special case unidirectional data transfer mode. Further, the new TCP header is disregarded by each of the end point receivers once the true TCP/IP header is reconstructed as shown in FIG. 5. Lastly, full 20 byte TCP/IP headers are transmitted only upon startup. For the steady state case, which is other than these two events, the typical 20 byte TCP/IP header has been shortened to a mere four bytes.

Referring next to FIG. 8 the methodology 170 for the mobile station compression for new TCP header 160 is shown in flowchart form. The mobile station process 170 is begun for the header compression and the start block enters block 171. Block 171 determines whether the present data packet is the first data packet for the call. If the packet is the first data packet block 171 transfers control to block 172 via the yes path. Block 172 sends the complete UDP header. For all other voice data packets the new TCP header 160 which comprises four bytes is sent along with the voice data to the receiving end. Again, the Van Jacobson method 20 is guided by the ancillary TCP header 30 to believe that a unidirectional data transfer is being sent with the header marked as described by byte 161.

Next, block 173 sends a complete RTP header only for a first voice data packet. Again for all other voice data packets the compressed header 160 is transmitted by ancillary TCP header 30.

Then, block 174 creates the new or ancillary header 160. Next, block 175 sends the complete TCP/IP header. This is the typical TCP/IP header 140 shown if FIG. 5.

If the present data packet is not the first data packet of the call, then block 171 transfers control to block 176 via the no path. Block 176 compresses the UDP/RTP header to 3 bytes as discussed for block 10, above. Next, block 177 sends the created new header to block 20 as a unidirectional transfer special case. The ancillary or new TCP header is produced which includes the bit pattern shown in byte 161. This particular bit pattern indicates to the Van Jacobson method 20 that it is a special case which is a unidirectional data transfer and bypasses the error mechanisms of the Van Jacobson methodology 20. The UDP checksum is stored in TCP checksum for further compression of the RTP/UDP header. Lastly, block 178 sends the four byte ancillary header with one byte of the three byte compressed RTP/UDP header to PDSN 55.

The mask byte 161 indicates the particular special case for unidirectional data transfer, that is, the rightmost four bits set equal to one. Transmit compression process 170 is then ended.

On the receive end or decompression the mobile station receive process 180 is performed. Again, both mobile station and the network PDSN must include a receive process and a transmit process.

The process is started and block 181 is entered. Block 181 determines whether the present data packet is the first data packet of the call. If it is the first packet, then block 181 transfers control to block 182 via the yes path. Block 182 stores all the fields of the UDP header as shown in FIG. 2. That is, all the fields which are cross hatched in FIG. 2 are stored by the mobile station. For the UDP header 110, these fields are the source, destination port and length.

Next, block 183 stores all the non-recurring information included in the RTP header of the first voice data packet. This includes the information in the RTP header 120 which is cross hatched in FIG. 3; namely, the PT field and the synchronization source identifier.

Next, the mobile unit stores all the non-recurring information included in the TCP/IP header of the first voice data packet, block 184. The fields of TCP/IP header 140 that are stored include the protocol version, header length, type of service, DF field, MF field, fragment offset, time to live field, protocol, source address, destination address, source port, destination port, data offset 1 and 2, A field, R field, S field and F field. These fields are the cross hatched fields shown in FIG. 5. Then, block 184 creates the ancillary header. Lastly, block stores all the fields in an IP header.

If the present data packet is not the first data packet, then block 181 transfers control to block 186 via the no path. Block 186 receives the seven byte header (RTP/UDP-TCP/IP).

Block 187 regenerates the 40-byte TCP/IP header shown in FIG. 5. Block 188 then discards the ancillary or new TCP header.

Block 189 then regenerates the RTP/UDP header which is 20 bytes. The UDP header 110 is shown in FIG. 2. The RTP header 120 is shown in FIG. 3. The process 180 is then ended.

FIG. 10 depicts the compression process 190 for use by the PDSN 55. Process 190 is started and block 191 is entered. Block 191 determines whether the present block is the first IP packet of the call. If it is the first packet, block 191 transfers control to block 192 via the yes path. Block 192 stores all the fields of the uncompressed TCP/IP header. Then block 193 sends a complete TCP/IP header and process 190 is ended.

If the present data packet is not the first data packet, then block 191 transfers control to block 194 via the no path. Block 194 uses the Van Jacobson method to compress the 40 byte TCP/IP header to the new or ancillary 4-byte header shown in FIG. 7. The unidirectional transfer special cases is used by Van Jacobson TCP/IP header compression 50 to accomplish this. Lastly, block 195 sends the new or ancillary TCP/IP header to a mobile station. The process is then ended.

FIG. 11 depicts the decompression process 200 for use by the PDSN 55. Process 200 is started and block 201 is entered. Block 201 determines whether the present block is the first IP packet of the call. If it is the first packet, block 201 transfers control to block 202 via the yes path. Block 202 receives the 40-byte uncompressed TCP/IP header. Then block 203 sends the complete 40-byte uncompressed TCP/IP header and process 200 is ended.

If the present data packet is not the first data packet, then block 201 transfers control to block 204 via the no path. Block 204 receives the new or ancillary 4-byte header shown in FIG. 7. Lastly, block 195 uses the unidirectional data transfer special cases of by Van Jacobson TCP/IP header compression 50 to regenerate the complete TCP/IP header of FIG. 5. The process is then ended.

A full 40 bytes of headers is still sent initially. 20 bytes are the combined UDP/RTP headers and 20 bytes are the

TCP/IP headers. For voice data packets other than the initial, the four byte compressed TCP/IP header 160 is sent by the transmitting end and full TCP/IP header is reconstructed at the receiving end. Thus for the great majority of voice data packets 16 bytes of the 20 byte TCP/IP header are saved. This greatly enhances RF link utilization and bandwidth. This represents approximately an 80% savings in header overhead information which is required to be transmitted from the receiving to transmitting ends.

Although the preferred embodiment of the invention has been illustrated, and that form described in detail, it will be readily apparent to those skilled in the art that various modifications may be made therein without departing from the spirit of the present invention or from the scope of the appended claims.